Hardware/Software Security Patches for the Internet of Things

John A. Stankovic[1] Tu Le[1] Abdeltawab Hendawi[2] Yuan Tian[1]

[1]*University of Virginia, VA, USA*
stankovic@cs.virginia.edu, tnl6wk@virginia.edu, yuant@virginia.edu
[2]*University of Rhode Island, RI, USA*
hendawi@uri.edu

***Abstract*—With the rapid development of the Internet of Things (IoT), there are billions of interacting devices and applications. With so many devices and applications, one of the most critical challenges is how to provide security. Traditional software-based defenses will not be enough to protect the security of IoT because of the attack surfaces derived from the physical environment. For example, an attacker can physically re-point a surveillance camera, can move a smart device to another location, can send a sound signal to influence an accelerometer, can cause wireless jamming, etc. We propose to create "smart buttons," and collections of them called "smart blankets" as hardware/software (HW/SW) security patches rather than software-only patches. These fixes operate similarly to software patches, but because of the hardware added, these new patches can better support against physical world attacks. While this paper primarily presents a vision for HW/SW patches, solutions are implemented and shown for two classes of attacks involving cameras and robots. Open questions are also discussed.**

I. Introduction

The Internet of Things (IoT) market estimates that there are around 30.7 billion IoT devices currently in use. This number is projected to grow to 75.4 billion devices by 2025. At such a rapid pace, more and more newly connected devices, as well as applications such as autonomous vehicles or robots, smart homes, and smart cities are appearing on the Internet. Applications will both, directly and indirectly, interact with users, devices, other applications, and the environment. For example, in smart home environments, there already exists IoT devices such as security cameras, smart locks, garage door openers, smart thermostats, smoke detectors, and they are all connected on the Internet. There are also Industrial IoT systems where perhaps thousands of devices are monitoring and controlling a process control plant, and robots are used to deliver basic resources around the plant. Smart cities are also seeing an ever-increasing deployment of sensors, actuators, and services to monitor and control transportation, emergency services, pollution, energy, health, etc.

Once large IoT systems are installed, they will potentially exist for a long time and are difficult to shut down and reinstall updates. Unfortunately, these systems are often subject to security attacks. Further, because of the physical nature of IoT, software patches alone are not always sufficient to react to the attacks. These new *physical* challenges for security in IoT are due to many factors. For example, attacks generally span across three layers: the physical, communication, and application layers. For the physical layer, the fact that the smart devices exist in open environments exposes them to tampering. Attackers can also attack the devices indirectly by changing the surrounding environment that the devices are monitoring. Communications are mostly via various wireless technologies, allowing easy access to devices and applications. The great heterogeneity among devices and communications complicates security solutions. Applications and services will often include software that is installed on the device to control it.

As a new direction, we introduce a new type of security patch that combines hardware and software (HW/SW). The goal for this device, called a "smart button," is to improve response to security attacks for IoT, especially because of the physical limitations of these systems. The main goals of this paper are to present and demonstrate solutions for smart buttons and to introduce directions and open issues for smart blankets. Smart blankets are collections of smart buttons that can be considered a type of security HW/SW middleware targeted towards security protection at scale. Detailed solutions for smart blankets are beyond the scope of this paper.

It is important to note that each IoT system will have its own security measures (e.g., redundancy, including possible sensor redundancy, secure keys, encryption, and blockchains). We are not proposing ideas to build a secure IoT application in the first place, but to "cover" it with our smart buttons and blankets when unanticipated attacks occur and **when standard software-only patching techniques fail to work**, or it is not feasible to replace devices and reinstall the software with the new fixes. Of course, future systems may incorporate HW/SW patch ideas into their systems in the first place.

In this paper, we make the following contributions:

- We propose a new concept and design of HW/SW security patches, called smart buttons, for the IoT.
- We build and show the utility of two experimental prototypes of HW/SW security patches, one involving IoT cameras and another involving robots.
- We provide insights and open questions into protecting the IoT with smart blankets as larger-scale attacks occur.

II. Related Work

There exists significant research on security threats on IoT. This work can be classified into three main categories: hardware, communications, and applications.

Hardware: IoT devices are widely used for different purposes, such as for smart homes and smart manufacturing. In reality, there is a critical need to develop careful security assurance methods from the ground-up, with IoT security in mind [2], [3]. Failure to protect IoT devices from security threats can lead to severe consequences. The diversity of device features, make the attack surface in IoT devices larger and more complex than in other systems. Researchers presented vulnerabilities of existing smart locks that would allow attackers to gain sensitive user information and unauthorized access to a smart home [8]. Due to weak password or unprotected debugging interfaces, many Telnet-capable IoT devices are considered to be vulnerable [21]. Other studies have shown that proper protections against malicious attacks are missing for sensors largely used in the smart devices [6], [14], [22].

Communications: Integration and communication between IoT devices themselves as well as with the public internet introduce a wide surface for security threats [18]. Hong et al. showed that desired network communications must be explicitly enabled for communication among IoT devices for better security [9]. Some complex network attacks include interrupting jamming, scan jamming, and pulse jamming are discussed in [1]. Sivaraman et al. proposed a three-party architecture for monitoring network activity to detect unusual behavior [16]. Once an attacker succeeds in obtaining access into the network of IoT devices with the help of a malicious device, other devices connected to the same network could be easily threatened [11].

Applications: IoT programming frameworks enable developers to build applications to deal with common types of devices such as motion sensors, fire alarms, and door locks [5]. However, those frameworks also give malicious users the ability to cause major security risks to the underlying IoT devices. There has been some recent research on the security analysis of smart home applications. Fernandes et al. analyzed Samsung's SmartThings which has the largest number of applications among currently available smart home platforms, and presented an attack where malicious applications can steal sensitive information such as lock codes of the IoT devices [5].

The proposed smart button/blanket can help the underlying IoT devices to identify the attacks at different levels and react to those attacks. The smart button/blanket will add extra sensing hardware (e.g, accelerometer, gyroscope, GPS, temperature, compass etc.), and software patches to detect if the to-be-protected IoT device has been physically moved to another location, changed its direction or behavior, tampered, or controlled by a remote attacker. An appropriate reaction (e.g., send an alarm, shut down the IoT device, correction) will be taken by the smart button, in case an attack is discovered.

III. Proposed Solution: Smart Button

In this section, we present our proposed smart button as hardware and software patches for added security of IoT.

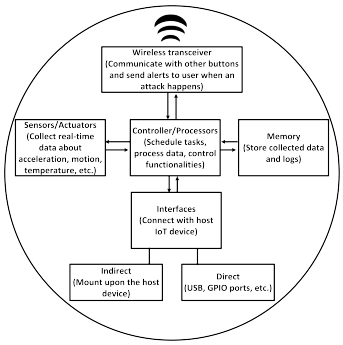


Fig. 1: Overview of the smart button building blocks.

A. Security Attack Models

The IoT has an extremely broad and diverse attack surface, and it is changing all the time. New attacks are expected to appear over time, i.e., the proverbial arms race in computer security. There have already been many attacks that often remain undetected or unpatched over a long time. For example, the railway and electricity power industries in Ukraine were attacked by BlackEnergy crimeware [4], and nuclear uranium facilities were attacked by Stuxnet [10].

For IoT, we can consider security attacks at three layers: physical, communication, and application. This paper primarily focuses on physical attacks. Physical attacks: include that an attacker can physically re-point a camera, can tamper with a device such as covering a sensor; can set up actions in the environment to fool sensors, e.g., broadcast the sound of a voice to activate an app when not appropriate, can use side channels based on resonance [6], [7], and can trick sensors, delay/speed up / distort data in real-time streams. For more information on attack models see [17].

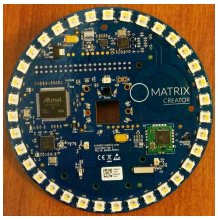
B. Smart Button Building Blocks

An HW/SW patch (a smart button - See Figure 1) [17] combines sensors, actuators, processors, and memory with associated software to protect IoT devices against one or more specific security attacks. In some cases, it also includes wireless communications to report the attack. It is meant to be used in IoT systems where the physical world and smart devices are key components. It is important to note that software patches alone cannot provide a redundant sensor or a correlated sensor modality to support a better security solution.

To date, we have built a prototype of the smart button and used it for several attacks. Our smart button basically consists of a Raspberry Pi and a set of sensors connected to it. For a general prototype design, we used a Raspberry Pi 3 Model B+ and a Matrix Creator board (see Figure 2). These two components interface with each other through GPIO pins. The Matrix Creator includes different types of sensors (e.g., light sensor, accelerometer, gyroscope, and temperature), wireless

communications, and an FPGA. Matrix Hardware Abstraction Layer (HAL) is an open source library that consists of C++ drivers, allowing access to the components of the Matrix Creator. Using Matrix HAL, we created a program for the Raspberry Pi to continuously take sensing data from the Matrix Creator and perform computations. It is important to note that our smart button building blocks can be flexible, depending on the host device and the security requirements.

(a) Raspberry Pi



(b) Matrix Creator

Fig. 2: Example of a Smart button generic prototype.

C. Protecting IoT at Various Layers

Before presenting the implementation and experiments (in the next section) with the smart button, in this section we briefly describe how smart buttons can protect the IOT at the physical, communication, and application layers.

1) Physical Layer: Our approach for the physical layer is to create a set of modularized, extensible smart buttons designed to provide generalized security features related to different attack types on the physical world and physical devices. The buttons themselves will support button-button and button-IoT node interaction using interfaces that can be updated in future HW/SW revisions, supporting back compatibility. We assume that buttons and blankets interact with a higher layer "hub" that is much more capable and OTA upgradeable (note that communication channels are assumed to exist, as indicated by the IoT deployment, but their security is itself a target for attack and protection by blankets). It is not possible to exhaustively address all attack types, so instead it may be necessary to characterize broader attack categories, and then demonstrate physical layer button solutions that both address the specific attack example and also portray the flexibility of the button attributes. For example, smart buttons can provide physical layer redundancy to an IoT system to detect attacks by flagging undesirable behavior, regardless of the specific attack source. For example:

1) Perform redundant sensing to detect alterations to sensed values, such as adding a temperature-based button to measure temperature in the same environment as existing IoT temp sensors.
2) Perform sensing with complementary modality to IoT sensor - same purpose as above.
3) Detect movement of a device that is not supposed to move (e.g., if someone re-points a fixed video camera).
4) Detect tampering to physical packaging (e.g., by sensing motion or sound, or by measuring changes to the response of a package to a stimulus as used in structural monitoring).
5) Detect erroneous physical world signals injected by attackers.

Upon detection of a potential security attack, the button or blanket, if deployed, must sometimes notify other buttons or blankets and higher layers to enable Response and Recovery actions. Other sensor-based types of buttons can thus offer the functions to further protect existing deployments (of IoT nodes and/or previously deployed blankets) against or respond to the results of both known and unknown attack models.

2) Communication Layer: Since the IoT will rely heavily on wireless communications, addressing security attacks in this area are paramount, including those in Table 3.

Network layer	Attacks	Defenses
Physical	Jamming	Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change
	Tampering	Tampering-proofing, hiding
Link	Collision	Error-correcting code
	Exhaustion	Rate limitation
	Unfairness	Small frames
Network and routing	Neglect and greed	Redundancy, probing
	Homing	Encryption
	Misdirection	Egress filtering, authorization, monitoring
	Black holes	Authorization, monitoring, redundancy
Transport	Flooding	Client puzzles
	Desynchronization	Authentication

Fig. 3: Sensor network layers and denial-of-service defenses.

In general, it is necessary to consider communications attacks on both the original IoT systems and on the smart buttons and blankets. Capabilities can be added to smart buttons to support attacking the attacker and to create notifications.

For our solution at the communications layer, the general capabilities based on the redundancy of wireless devices and supportive software (i.e., a hardware/software patch) of the buttons include:

- Overhearing.
- Ability to detect jamming of various types.
- Use of multiple frequency channels.
- Capability to provide burst alarms.
- Continue to operate when jamming occurs.

Details on these solutions are outside the scope of this paper, but the solutions found in the paper [20] can be added to a collection of smart buttons.

3) Application Layer: IoT systems will provide many smart services. Big Data and associated machine learning and analytics will be a core ingredient of these systems. In these situations smart buttons/blankets can provide an extra layer of security by avoiding bad data being passed to the application layer. Note that much of the data used by applications and its processing will reside in the Cloud or other servers. We are not proposing to improve the security of the Cloud or internal security mechanisms of the smart services themselves. Rather, our approach for applications focuses on the added data protections that buttons and blankets can provide.

For example, since we are dealing with the physical world, physical attacks can affect the application layer such as when

attackers modify camera pixels of a stop sign to fool a self-driving car's image processing algorithm, thereby causing a crash.

IV. Attack Scenarios and Defense Results

In this section, we describe the physical layer results of using our smart button solution to address two attack scenarios: fixed-position camera re-positioning and LiDAR attacks on robots.

A. Fixed-position Surveillance Camera

Consider a fixed-position surveillance camera. Assume that a smart camera was installed in the front door of a house, and it is supposed to remain fixed in position. In this case, any movement of the camera that is not made by the legitimate user, such as physically re-pointing or relocating the camera, is considered malicious. To detect this attack, it may be possible to install a software-only patch on the camera that relies on the camera's capabilities such as image and video capture of background and detecting that the background is moving to detect attacks. However, leveraging such capabilities requires a lot of computational power that the camera may not have enough resources to provide or the internals of the camera software may be proprietary. In this scenario, if a smart button is used (e.g., adding an accelerometer and associated software to detect movement and send an alarm if necessary), there are no changes to the camera, and this attack can be detected. The button provides acceleration sensing capability from an accelerometer, which the camera does not have in the first place, to detect any malicious movement of the camera.

Our smart button prototype for this scenario consists of a Raspberry Pi connecting with a Matrix Creator board (see Figure 2). The two components interface with each other through GPIO pins. The Matrix Creator includes different types of sensors (e.g., light sensor, accelerometer, gyroscope, and temperature), wireless communications, and an FPGA. However, to address this scenario, our smart button only needs to consider the acceleration readings.

Since the button is attached to the camera, re-positioning the camera will lead to the motion of the button. We write a function for the button that utilizes the acceleration readings from the Matrix Creator's accelerometer to detect motion. This program acts as a utility service that continually runs in the background. If there is an abnormal change in the acceleration readings that exceed the threshold value set by the user, the button will consider it to be a re-position event. For this experiment, we first place the prototype in a stationary position and activate the program. We then re-point the prototype in a different direction. As a result, we get an alarm sent from the button to our server, notifying that there has been a re-position.

Note that this solution is representative of many similar situations. For example, there can be many other attacks that move a smart sensing device or that corrupt its sensors. Here we can apply the same strategy except we utilize redundant or complementary sensors such as temperature, vibration, gyroscopes, etc. As one example, if an attacker has infiltrated the control software for keeping the temperature of a chemical process below a threshold, that attacker can cause overheating. A new HW/SW patch that is independent of the previous system can be added that monitors temperature or even monitors a complementary metric such as pressure can detect this attack.

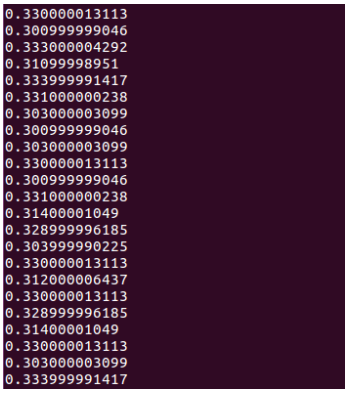
B. Robot with LiDAR-based Navigation

Even without special-purpose equipment, an adversary can easily exploit physics to manipulate the outputs of sensors, causing unexpected behaviors of the systems that rely on those sensors [6]. LiDAR has been shown to be vulnerable to physical attacks that exploit the characteristics of light sensing [12], [15]. Petit et al. [12] show that they could generate fake dots that are far away in the laser scan readings of the IBEO Lux 3. As a follow-up work, Shin et al. [15] presented LiDAR saturating attack and an improved version of spoofing attack on the Velodyne VLP-16.

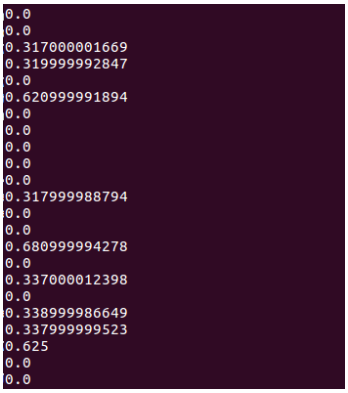
In this experiment, we employ the TurtleBot3 robot model equipped with a 2D laser scanner LDS-01 [13]. We consider the scenario that an adversary physically attacks the LiDAR on a robot to compromise its laser scan readings to affect the robot's behaviors, such as obstacle avoidance or navigation planning. We first craft an attack on the LiDAR of the robot. Then, to launch the attack, we use another robot of the same model to act as an attacker. As a result, there are abnormal values in the front angle readings of the victim's LiDAR when the attack happens (see Figure 4).

We design a prototype of the smart button with LiDAR sensing capability to provide redundant sensing to the robot. Our smart button with separate LiDAR readings helps to detect the abnormal laser scan readings caused by the attack on the original LiDAR of the robot. Our smart button prototype for this scenario consists of a Raspberry Pi connecting with a LiDAR. The two components interface with each other through a USB interface. The LiDAR sensor readings are given as an array of distance-to-obstacle values. The array's indexes are scan angles of the LiDAR, and the values are the distances to the obstacles (if any). Let L be the array of readings from the LiDAR and α be the scan angle at which the LiDAR detects an obstacle. We have $L[\alpha]$ as the distance to the obstacle at angle α. Note that we need to calibrate the readings because the original LiDAR of the robot and the smart button are different in position. To be more specific, the smart button is placed next to the original LiDAR, pointing in the same direction (see Figure 5). In our robot system with the smart button attached, we have $L1[\alpha]$ as the distance to the obstacle at angle α measured by the robot's original LiDAR (LiDAR 1) and $L2[\beta]$ as the distance to the obstacle at angle β measured by our smart button (LiDAR 2). We also have d as the distance from LiDAR 1 to LiDAR 2. We can calculate β, d2 from α, d1, and d. Then, we compare the calculated β, d2 with the actual sensor readings of our smart button. When there is a noticeable mismatch between readings of two LiDARs, the button detects an attack.

For this experiment, we first implement a LiDAR-based navigation module, which includes obstacle avoidance, for



(a) Normal Scenario



(b) Attack Scenario

Fig. 4: Front angle LiDAR readings in normal and attack scenarios. Note that the surrounding environment remains the same in both scenarios.

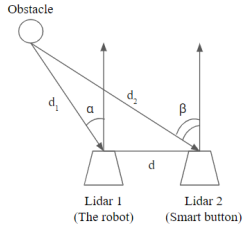


Fig. 5: Redundant sensing for LiDAR using smart button.

the benign robot and have it move in a predefined path. The benign robot is expected to recognize obstacles and navigate to avoid them. We then place the attacker, which is another robot with the same model, at a point in this path. When the benign robot moves to this point, it crashes into the attacker. To defend against this attack, we rerun the experiment with our smart button prototype installed onto the benign robot. As a result, the benign robot was able to recognize and avoid the attacker. We also get an alarm sent from the button to our server, notifying that there has been an attack.

V. Discussion

A. Interfacing

A key research challenge is interfacing the smart buttons to the IoT devices and to each other. The challenges vary depending on the situation. While open questions remain and are beyond the scope of this paper, to illustrate the complexity we briefly describe 5 categories of interfaces. See [17] for more information.

Indirect: In some situations a button will be mounted upon a ”smart thing/device” and need only monitor it in indirect ways. For example, a smart button can be attached to a smart camera that has fixed orientation and the smart button can monitor if the camera is moved (via an accelerometer in the smart button).

Direct without much knowledge: Since a button is an HW/SW entity, it can attach to IoT devices and applications in various ways. For situations where the button designer is not aware of the details of devices and/or applications, we propose to treat the smart button as an I/O device with a driver. The driver may be specific to a given IoT device and/or OS, so this abstraction allows buttons to connect to any existing device with a known interface with minimal new development beyond the driver that bridges the button API to the existing interface.

Direct with knowledge: In some situations, the smart buttons deployment team might be aware of the internals of the smart IoT devices. If the smart button is aware of these details and can directly access the data (via a direct with knowledge interface), then more sophisticated security detection and actions can be possible.

Button to button: Since smart buttons are created by the security team, their interfaces to each other can be standardized. An open challenge is to develop a button-to-button communication interface with the following characteristics: wirelessly communicates, can reach at least 100 feet, encrypts data and control signals during communication, can utilize different frequencies, can detect and mitigate jamming, and supports transferring (exchanging) of security intrusion detection, attack information, and blanket control commands.

Blanket to blanket: Our hypothesis is that once a smart blanket is added to a system, it can be considered as an integral part of the original system, so adding yet another blanket should be no different than adding the first. However, various optimizations may be possible across blankets.

B. Future Work

Designing smart buttons is one step towards HW/SW solutions to protect the IoT. Many interesting research questions remain. In the following, we discuss some challenges and future directions.

Collections of smart buttons forming the smart security blanket for a given security attack will communicate among themselves (with anti-jamming support) and offer various redundancies and diversity modalities. For example, the buttons can communicate with each other with security properties similar to those discussed above for protecting the communications of IoT services. They can form a consensus on both sensor readings and control actuations by the redundancy they provide. They can also use alternative sensing (orthogonal) modalities to avoid various physical attacks. Many consensus

schemes can be implemented. Blockchain [19] is an open, decentralized digital ledger technology that creates a secure way for the exchange of data. It is being applied to IoT at the application layer. It is necessary to investigate if a lightweight blockchain can be developed for smart blankets. Hashing, proof-of-work, and a consensus found in blockchains may be too costly in time and energy to be an effective smart blanket mechanism.

Scaling is a very difficult issue. We have developed some preliminary ideas. Solutions must be developed that do not require EVERY device (sensor/actuator) to have a button attached to it. For example, if there are 10,000 smart devices in the subway system of New York City and there is a new attack requiring smart buttons, solutions are needed that can protect the system WITHOUT attaching a button to EVERY one of the 10,000 devices. In this case new solutions must be created where individual buttons can protect sets of devices. One approach might be: (i) creating smart buttons that act as HW/SW patch hubs; the hubs (analogous to routing hubs) will aggregate information from sets of devices and detect and protect based on the cumulative properties of those devices, (ii) consider the criticality of a device when needing to install a button to protect it, and (iii) create smart buttons that act as part of the fog, typically executing on a more powerful machine to provide local processing rather than sending the information to the cloud.

VI. Conclusion

In this paper, we present a novel design of HW/SW patches to handle security attacks in the era of IoT. Our solution approach is based on adding integrated hardware and software patches as a security monitoring and protection layer to the existing devices (things). One important lesson learned from this work is that software-only patches can solve many security issues in IoT, but not all of them. Software-only patches cannot detect many security issues due to missing hardware capabilities in the device. Thus, we introduce the smart button and smart blanket solution to fill these gaps. Although being an effective solution, smart buttons and blankets still have many open research challenges such as protecting themselves from attacks, synergistic interaction between different types of buttons in the form of a blanket to protect against complex security scenarios, and smooth interfacing with existing IoT devices. With new solutions for these issues, security defenses for the future IoT can be significantly improved.

References

[1] L. Almon, M. Riecker, and M. Hollick, "Lightweight detection of denial-of-service attacks on wireless sensor networks revisited," in *Local Computer Networks (LCN), 2017 IEEE 42nd Conference on*. IEEE, 2017, pp. 444–452.

[2] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, pp. 283 – 294, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618307035

[3] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, Jun 2018. [Online]. Available: https://doi.org/10.1007/s41635-017-0029-7

[4] F-Secure, "BlackEnergy & Quedagh: The convergence of crimeware and APT attacks," Tech. Rep., 2014. [Online]. Available: https://www.f-secure.com/documents/996508/1030745/blackenergy{_}whitepaper.pdf

[5] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 636–654.

[6] K. Fu and W. Xu, "Risks of trusting the physics of sensors," *Communications of the ACM*, vol. 61, no. 2, pp. 20–23, 2018.

[7] Y. Fu, E. Bauman, R. Quinonez, and Z. Lin, "Sgx-lapd: thwarting controlled side channel attacks via enclave verifiable page faults," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2017, pp. 357–380.

[8] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*. ACM, 2016, pp. 461–472.

[9] J. Hong, A. Levy, L. Riliskis, and P. Levis, "Don't talk unless i say so! securing the internet of things with default-off networking," *IOTDI*, 2018.

[10] D. Kushner, "The real story of stuxnet," *IEEE Spectrum*, vol. 50, no. 3, pp. 48–53, March 2013.

[11] Y. Oren and A. D. Keromytis, "From the aether to the ethernet-attacking the internet using broadcast digital television." in *USENIX Security Symposium*, 2014, pp. 353–368.

[12] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[13] ROBOTIS, "Turtlebot3," 2019. [Online]. Available: http://www.robotis.us/turtlebot-3/

[14] N. Roy, H. Hassanieh, and R. Roy Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '17. New York, NY, USA: ACM, 2017, pp. 2–14. [Online]. Available: http://doi.acm.org/10.1145/3081333.3081366

[15] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 445–467.

[16] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home iot devices," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*. IEEE, 2015, pp. 163–167.

[17] J. A. Stankovic, T. Le, A. Hendawi, and Y. Tian, "Hardware/software security patches for internet of trillions of things," *arXiv preprint arXiv:1903.05266*, 2019.

[18] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[19] M. Swan, *Blockchain: Blueprint for a New Economy*, 1st ed. O'Reilly Media, Inc., 2015.

[20] A. D. Wood, J. A. Stankovic, and G. Zhou, "Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE, 2007, pp. 60–69.

[21] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. ACM, 2015, p. 5.

[22] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 103–117.